Assignnment 1

2024-04-07

Question 1

```
#load data
data_set <- read_csv("Data4168216.csv")
```

```
## Rows: 10000 Columns: 204
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (204): Y, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#split data into all data (all) and data without additional noise variables (set)
train_all <- data_set[1:5000, ]
test_all <- data_set[5001:10000, ]



train_all <- as.data.frame(train_all)
test_all <- as.data.frame(test_all)
set.seed(123)

train_set <- train_all[,1:7]
test_set <- test_all[, 1:7]
```

10-Fold Cross Validation for KNN, (limited predictors)

```
set.seed(123)

#split the total train data into 10 groups
ctrl <- trainControl(method = "cv", number = 10)

#test values of k from 1 to 15
k_vals <- 1:15
cv_results <- data.frame(k = k_vals, accuracy = numeric(length(k_vals)))

train_set$Y = as.factor(train_set$Y)

#for each value of k, create a model that is trained using k fold cross validation
# and calculate the accuracy of that model
for (i in seq_along(k_vals)) {
  knn_model <- train(Y ~ X1+X2+X3+X4+X5+X6, data = train_set,
                     method = "knn", trControl = ctrl, tuneGrid = data.frame(k = k_vals[i]))
```

```r
  cv_results[i, "accuracy"] <- max(knn_model$results$Accuracy)
}

#find the optimal k, k with highest accuracy
optimal_k <- cv_results$k[which.max(cv_results$accuracy)]
print(optimal_k)
```

```
## [1] 15
```

KNN with Optimal K (limited predictors)

```r
predicted_classes <- knn(train_set[, -which(names(train_set) == "Y")],
                         test_set[, -which(names(test_set) == "Y")],
                         train_set$Y, k = optimal_k)


accuracy <- mean(predicted_classes == test_set$Y)
print(accuracy)
```

```
## [1] 0.709
```

10-Fold Cross Validation For Lambda (limited predictors)

```r
set.seed(123)

x <- model.matrix(Y ~ X1+X2+X3+X4+X5+X6, data=train_set)
y <- train_set$Y


cv_fit <- cv.glmnet(x = x, y = y, family = "binomial", nfolds = 10)

best_lambda_set <- cv_fit$lambda.min

best_lambda_set
```

```
## [1] 0.00463002
```

Lasso Regression (limited predictors)

```r
x_test <- model.matrix(Y ~ ., data = test_set)

lasso_model <- glmnet(x=x, y=y, family="binomial", alpha = 1, lambda = best_lambda_set)
preds<- predict(lasso_model, newx = x_test, type ="response" )

predicted_classes <- ifelse(preds > 0.5, 1, 0)

accuracy <- mean(predicted_classes == test_set$Y)

print(accuracy)
```

```
## [1] 0.6678
```

10-fold cross validation for KNN (all predictors)

```
set.seed(123)

ctrl <- trainControl(method = "cv", number = 10)

k_vals <- 1:15
cv_results <- data.frame(k = k_vals, accuracy = numeric(length(k_vals)))

train_all$Y = as.factor(train_all$Y)

#10 fold cross validation for knn
for (i in seq_along(k_vals)) {
  knn_model <- train(Y ~ ., data = train_all, method = "knn", trControl = ctrl, tuneGrid = data.frame(k
  cv_results[i, "accuracy"] <- max(knn_model$results$Accuracy)
}

optimal_k <- cv_results$k[which.max(cv_results$accuracy)]
print(optimal_k)
```

```
## [1] 13
```

KNN With Optimal K (all predictors)

```
set.seed(123)
predicted_classes <- knn(train_all[, -which(names(train_all) == "Y")],
                         test_all[, -which(names(test_all) == "Y")],
                         train_all$Y, k = optimal_k)

accuracy_knn_all <- mean(predicted_classes == test_all$Y)
print(accuracy_knn_all)
```

```
## [1] 0.5686
```

1O fold Cross Validation for Lambda (all predictors)

```
set.seed(123)

x <- model.matrix(Y ~., data=train_all)
y <- train_all$Y


cv_fit <- cv.glmnet(x = x, y = y, family = "binomial", nfolds = 10)

best_lambda_all <- cv_fit$lambda.min

best_lambda_all
```

```
## [1] 0.01288331
```

Lasso Regression (all predictors)

```
lasso_model <- glmnet(x=x, y=y, family="binomial", alpha = 1, lambda = best_lambda_all )
```

```r
x_test_all <- model.matrix(Y ~ ., data = test_all)
preds<- predict(lasso_model, newx = x_test_all, type ="response" )

predicted_classes <- ifelse(preds > 0.5, 1, 0)

accuracy_lr_all <- mean(predicted_classes == test_set$Y)

print(accuracy_lr_all)
```

```
## [1] 0.6684
```

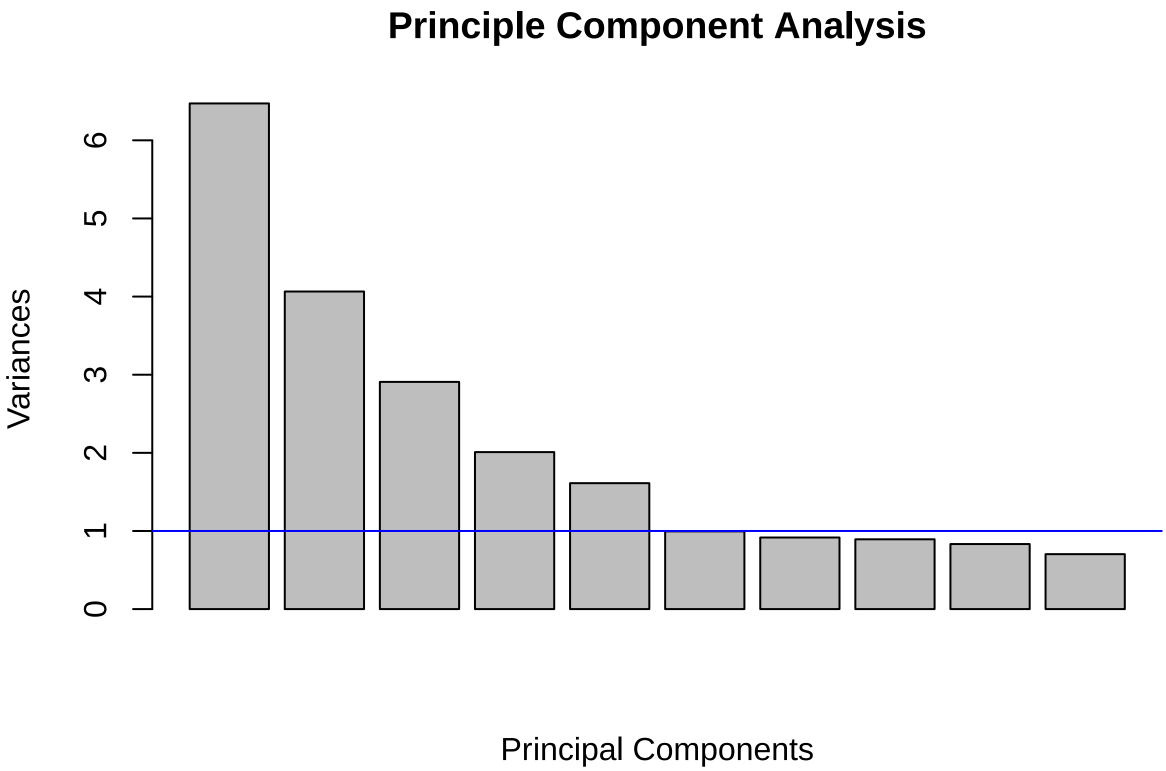
Question 2

Kaiser's Rule

```r
data_set <- read_csv("data.US.csv")
```

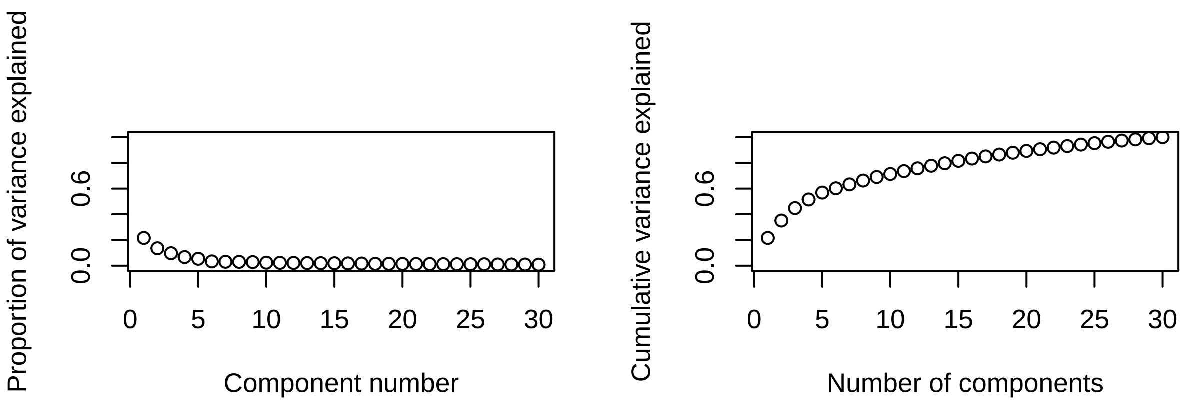
```
## New names:
## Rows: 1000 Columns: 31
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (31): ...1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```r
data_set = subset(data_set, select = -c(...1) )
pca_result <- prcomp(data_set, scale=TRUE)
plot(pca_result, xlab="Principal Components", main="Principle Component Analysis" )
abline(h = 1, col = "blue", lty = 1) #eigenvalues should be > 1
```



Since Kaiser rule states that only the principal components whose eigenvalues exceed 1 should be retained, the first 5 components are selected for PCA. the eigenvalue has to be more than 1, the first 5 components should be selected from PCA.

Scree Plots and Elbow Rule



Velicer's MAP test

```r
MAP(data_set, corkind='pearson', verbose = TRUE)
```

```
##
##
## MINIMUM AVERAGE PARTIAL (MAP) TEST
##
## Number of cases = 1000
##
## Number of variables = 30
```

```
##
## Specified kind of correlations for this analysis: Pearson
##
##
## Total Variance Explained (Initial Eigenvalues):
##             Eigenvalues   Proportion of Variance   Cumulative Prop. Variance
## Factor 1           6.47                     0.22                        0.22
## Factor 2           4.06                     0.14                        0.35
## Factor 3           2.91                     0.10                        0.45
## Factor 4           2.01                     0.07                        0.52
## Factor 5           1.61                     0.05                        0.57
## Factor 6           0.99                     0.03                        0.60
## Factor 7           0.92                     0.03                        0.63
## Factor 8           0.89                     0.03                        0.66
## Factor 9           0.83                     0.03                        0.69
## Factor 10          0.70                     0.02                        0.71
## Factor 11          0.67                     0.02                        0.74
## Factor 12          0.65                     0.02                        0.76
## Factor 13          0.61                     0.02                        0.78
## Factor 14          0.58                     0.02                        0.80
## Factor 15          0.56                     0.02                        0.82
## Factor 16          0.53                     0.02                        0.83
## Factor 17          0.50                     0.02                        0.85
## Factor 18          0.44                     0.01                        0.86
## Factor 19          0.43                     0.01                        0.88
## Factor 20          0.41                     0.01                        0.89
## Factor 21          0.40                     0.01                        0.91
## Factor 22          0.38                     0.01                        0.92
## Factor 23          0.35                     0.01                        0.93
## Factor 24          0.35                     0.01                        0.94
## Factor 25          0.33                     0.01                        0.95
## Factor 26          0.33                     0.01                        0.96
## Factor 27          0.28                     0.01                        0.97
## Factor 28          0.28                     0.01                        0.98
## Factor 29          0.27                     0.01                        0.99
## Factor 30          0.24                     0.01                        1.00
##
## Velicer's Average Squared Correlations
##    root   Avg.Corr.Sq.   Avg.Corr.power4
##       0        0.05902           0.00929
##       1        0.03580           0.00302
##       2        0.02673           0.00160
##       3        0.01860           0.00094
##       4        0.01351           0.00047
##       5        0.01229           0.00046
##       6        0.01340           0.00062
##       7        0.01535           0.00103
##       8        0.01752           0.00111
##       9        0.01924           0.00126
##      10        0.02187           0.00179
##      11        0.02543           0.00234
##      12        0.02869           0.00311
```

```
##      13       0.03281          0.00428
##      14       0.03743          0.00519
##      15       0.04339          0.00769
##      16       0.04984          0.01018
##      17       0.05602          0.01171
##      18       0.06475          0.01498
##      19       0.07336          0.01700
##      20       0.08436          0.02140
##      21       0.09947          0.02711
##      22       0.11009          0.03139
##      23       0.12879          0.04138
##      24       0.15808          0.05857
##      25       0.18966          0.08026
##      26       0.23855          0.12182
##      27       0.34499          0.21497
##      28       0.52087          0.39484
##      29       1.00000          1.00000
```

```
##
## The smallest average squared correlation is 0.01229
```

```
##
## The smallest average 4rth power correlation is 0.00046
```

```
##
## The number of components according to the original (1976) MAP Test is = 5
```

```
##
## The number of components according to the revised (2000) MAP Test is = 5
```

Horn's Parallel Analysis

```r
parallel_result <- paran(data_set)
```

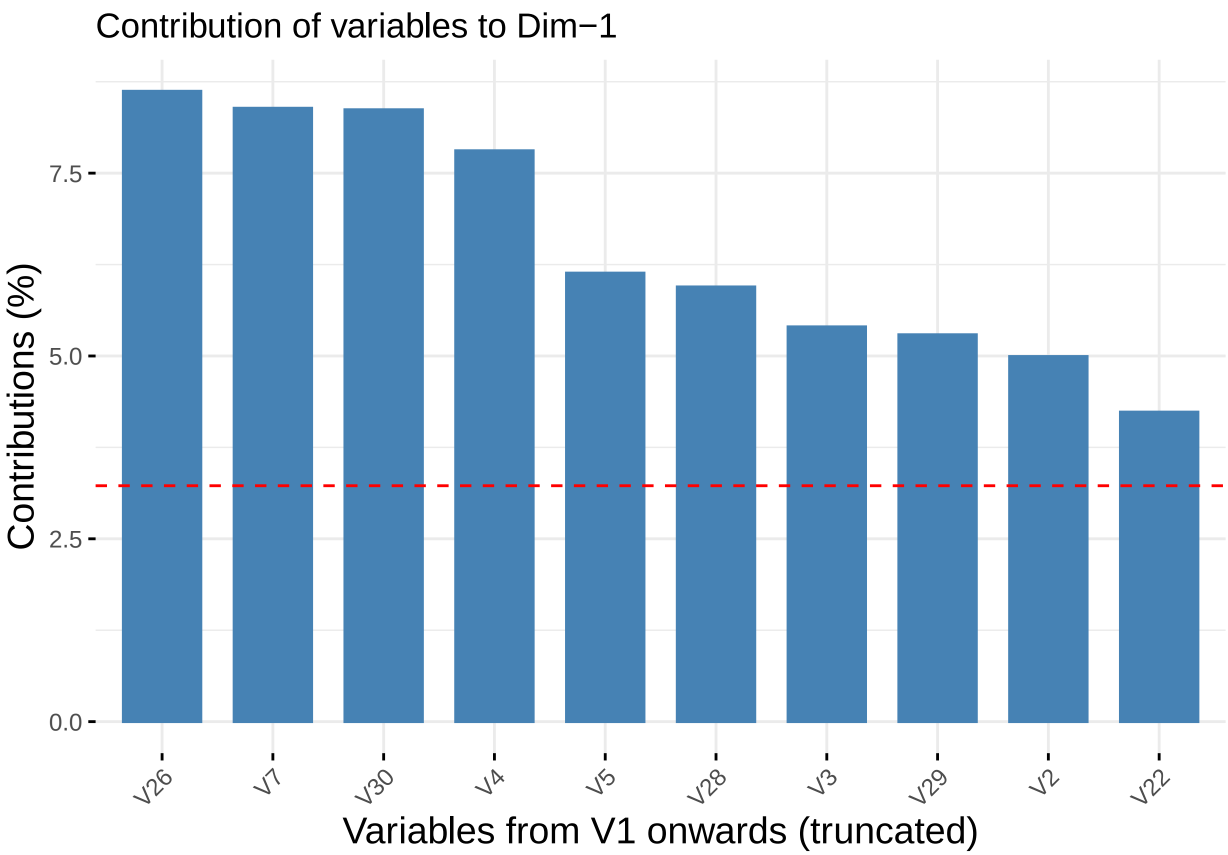
```
##
## Using eigendecomposition of correlation matrix.
## Computing: 10%  20%  30%  40%  50%  60%  70%  80%  90%  100%
##
##
## Results of Horn's Parallel Analysis for component retention
## 900 iterations, using the mean estimate
##
## --------------------------------------------------
## Component   Adjusted    Unadjusted    Estimated
##             Eigenvalue  Eigenvalue    Bias
## --------------------------------------------------
## 1           6.139309    6.472152      0.332842
## 2           3.775373    4.064775      0.289401
## 3           2.652127    2.908170      0.256042
## 4           1.782375    2.009196      0.226820
## 5           1.410573    1.611608      0.201035
## --------------------------------------------------
##
## Adjusted eigenvalues > 1 indicate dimensions to retain.
## (5 components retained)
```

Contributions of Each Variable to PCs

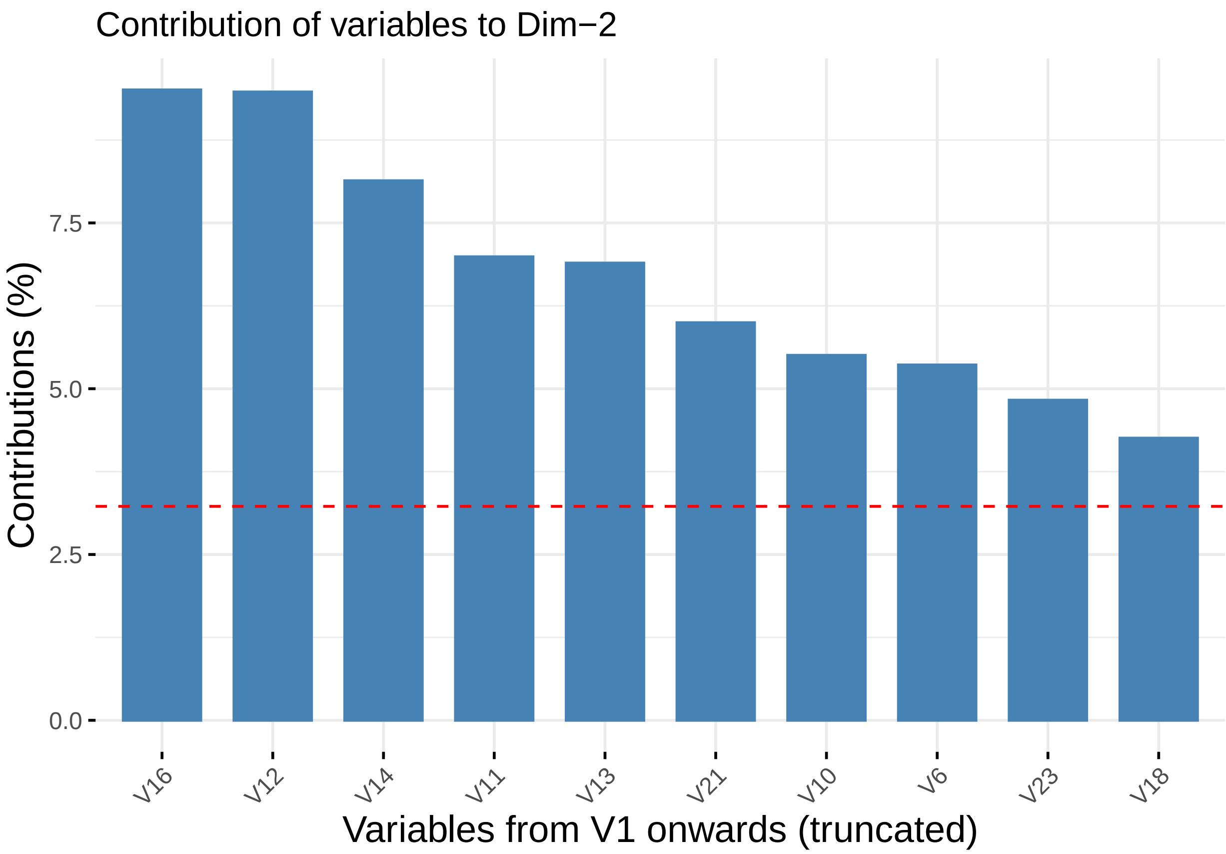
```
data_set <- read_csv("data.US.csv")
```

```
## New names:
## Rows: 1000 Columns: 31
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (31): ...1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

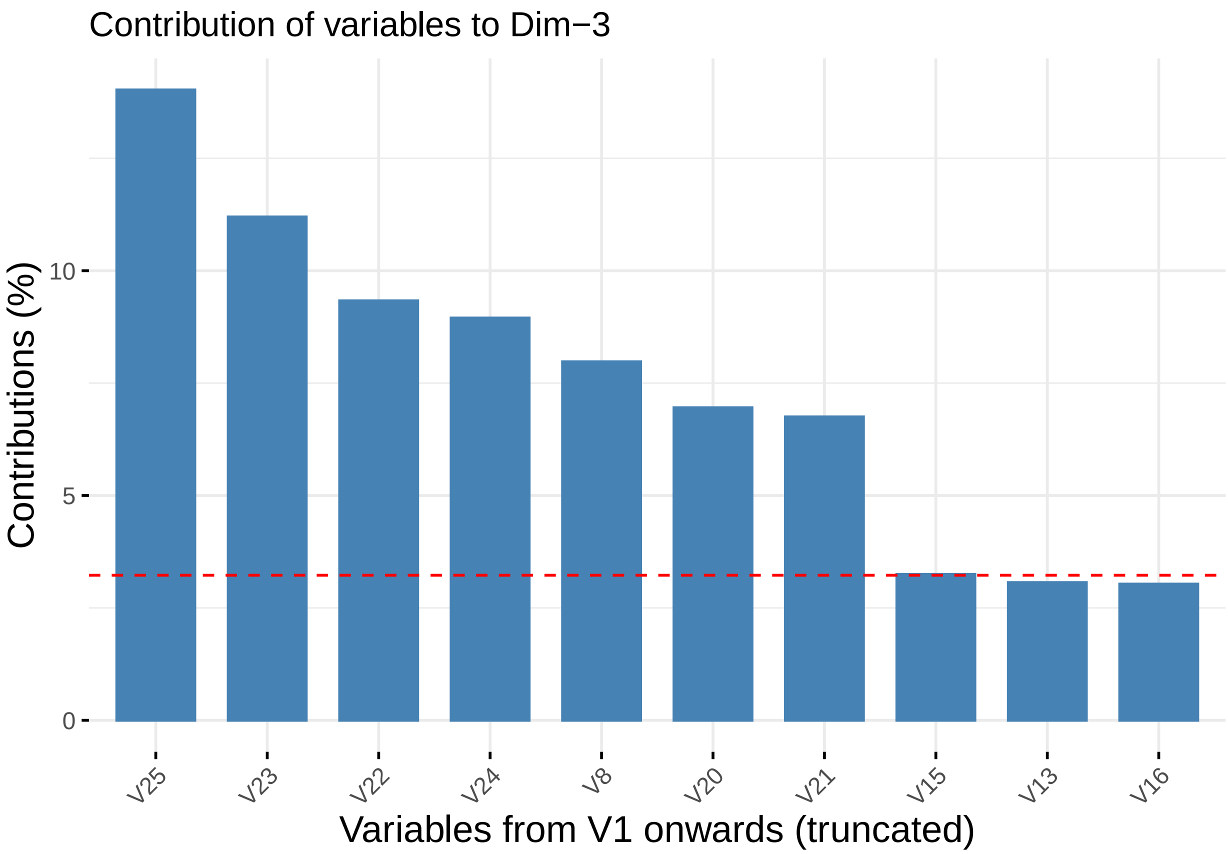
```
data.pca <- prcomp(data_set, scale = TRUE, center = TRUE)
var <- get_pca_var(data.pca)

# Contributions of variables to PC1
fviz_contrib(data.pca, choice = "var", axes = 1, top = 10) + xlab("Variables from V1 onwards (truncated
```



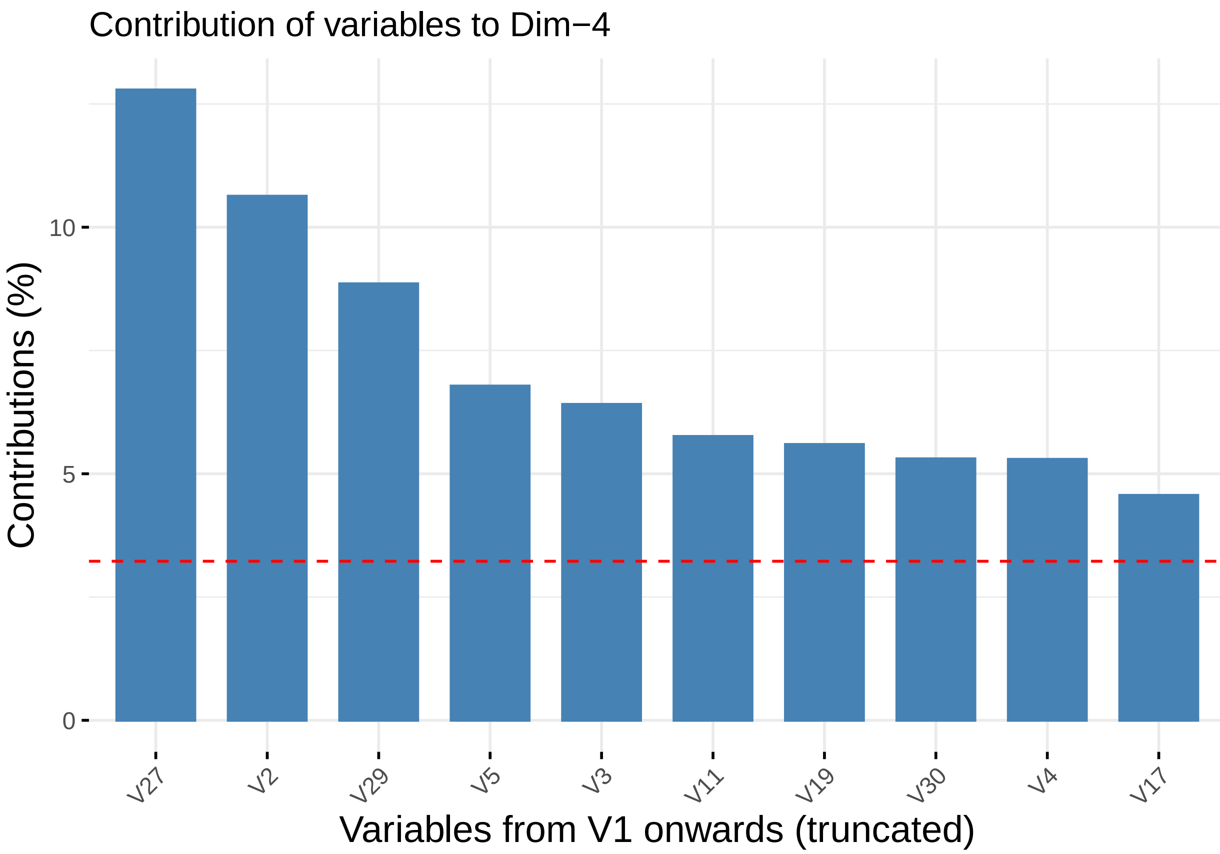

```
fviz_contrib(data.pca, choice = "var", axes = 2, top = 10) + xlab("Variables from V1 onwards (truncated
```



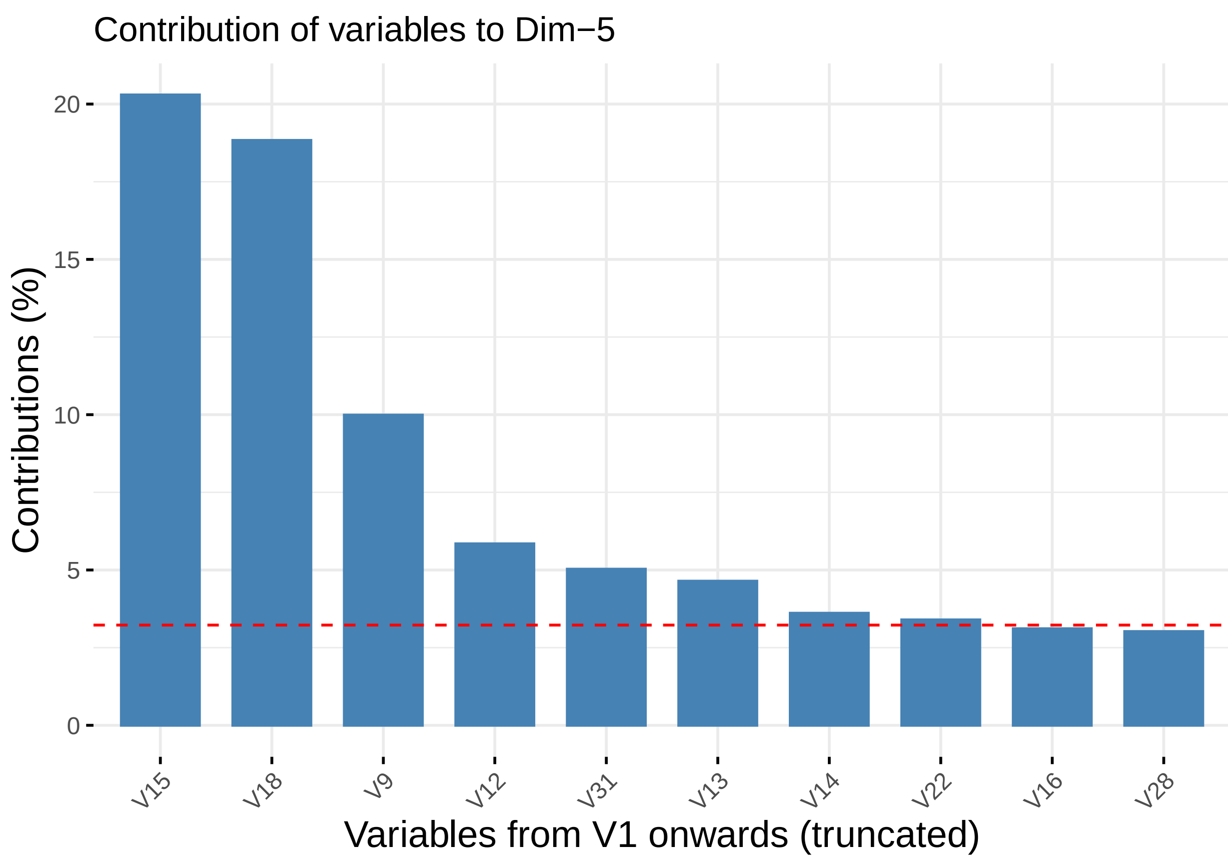

```
fviz_contrib(data.pca, choice = "var", axes = 3, top = 10) + xlab("Variables from V1 onwards (truncated)
```




```
fviz_contrib(data.pca, choice = "var", axes = 4, top = 10) + xlab("Variables from V1 onwards (truncated)
```




```
fviz_contrib(data.pca, choice = "var", axes = 5, top = 10) + xlab("Variables from V1 onwards (truncated)
axis.title.y = element_text(size = 14))
```

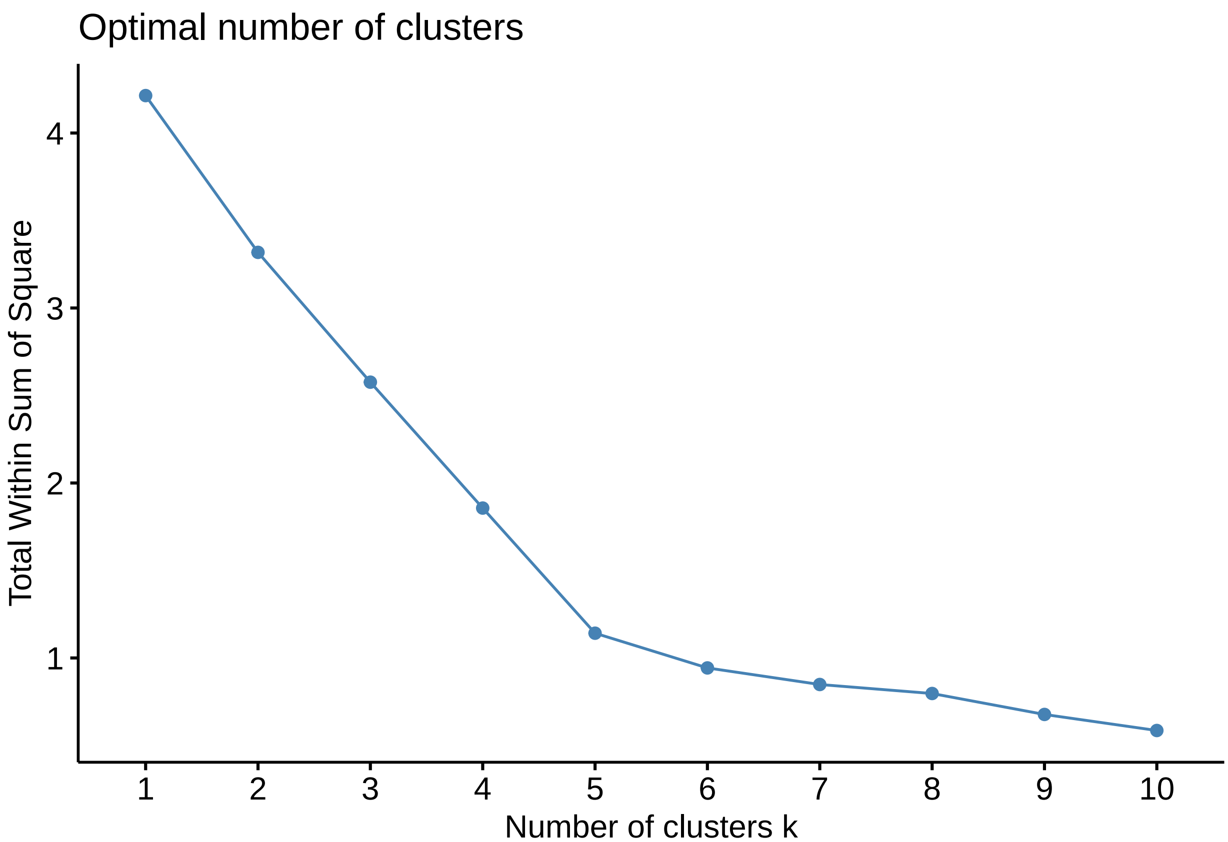



```r
summary(pca_result)
```

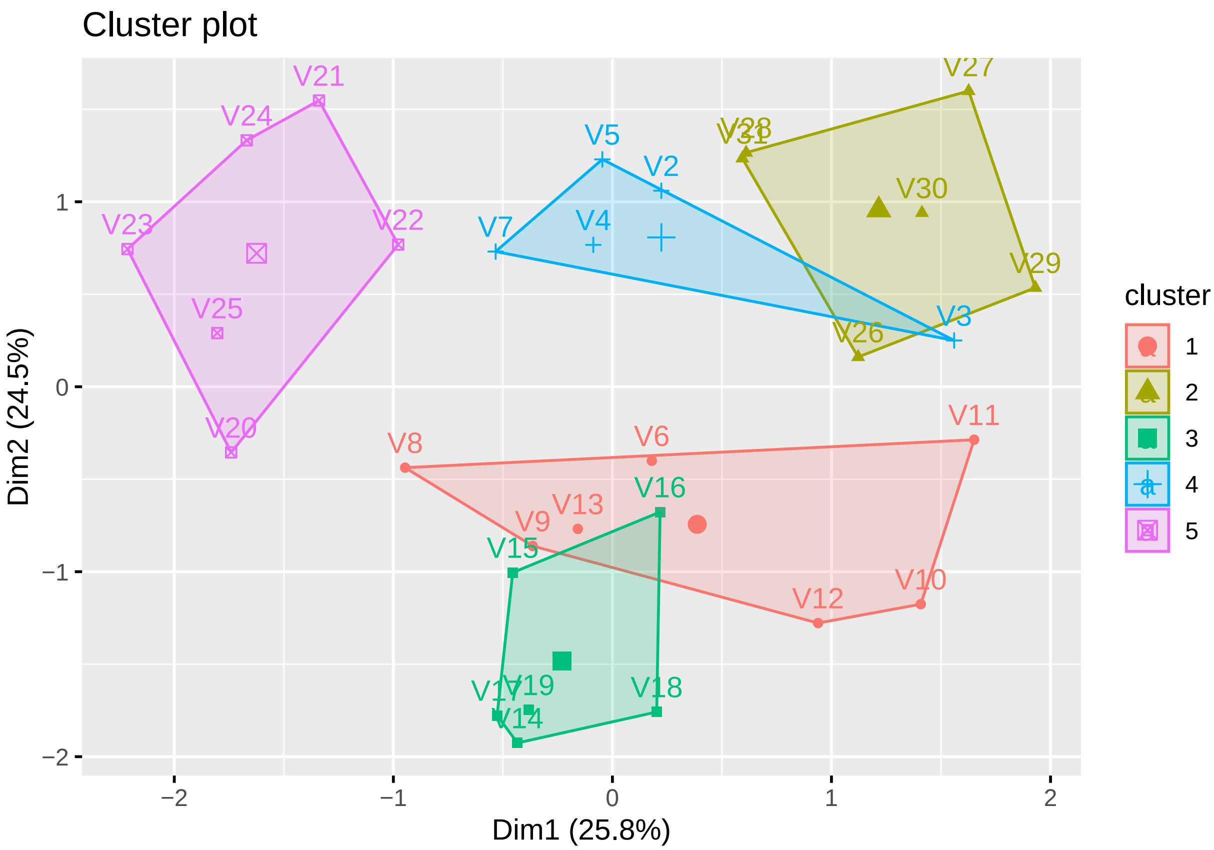
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.5440 2.0161 1.70534 1.41746 1.26949 0.99746 0.95706
## Proportion of Variance 0.2157 0.1355 0.09694 0.06697 0.05372 0.03316 0.03053
## Cumulative Proportion  0.2157 0.3512 0.44817 0.51514 0.56886 0.60203 0.63256
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.94509 0.91245 0.83845 0.81722 0.80452 0.78085 0.76348
## Proportion of Variance 0.02977 0.02775 0.02343 0.02226 0.02158 0.02032 0.01943
## Cumulative Proportion  0.66233 0.69009 0.71352 0.73578 0.75736 0.77768 0.79711
##                           PC15    PC16    PC17    PC18   PC19   PC20    PC21
## Standard deviation     0.75067 0.72702 0.70699 0.65981 0.6572 0.6433 0.62895
## Proportion of Variance 0.01878 0.01762 0.01666 0.01451 0.0144 0.0138 0.01319
## Cumulative Proportion  0.81589 0.83351 0.85017 0.86468 0.8791 0.8929 0.90606
##                          PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.6173 0.59481 0.59096 0.57732 0.57265 0.52971 0.52747
## Proportion of Variance 0.0127 0.01179 0.01164 0.01111 0.01093 0.00935 0.00927
## Cumulative Proportion  0.9188 0.93055 0.94219 0.95330 0.96424 0.97359 0.98286
##                           PC29    PC30
## Standard deviation     0.52003 0.49365
## Proportion of Variance 0.00901 0.00812
## Cumulative Proportion  0.99188 1.00000
```

K Means Clustering

```r
pca_data <- as.data.frame(pca_result$rotation[,1:5])

fviz_nbclust(x = pca_data ,FUNcluster = kmeans, method = 'wss' )
```




```r
k = 5
km.res <- kmeans(pca_data, centers=k, nstart = 25)

#print(km.res)

fviz_cluster(km.res, data=pca_data)
```

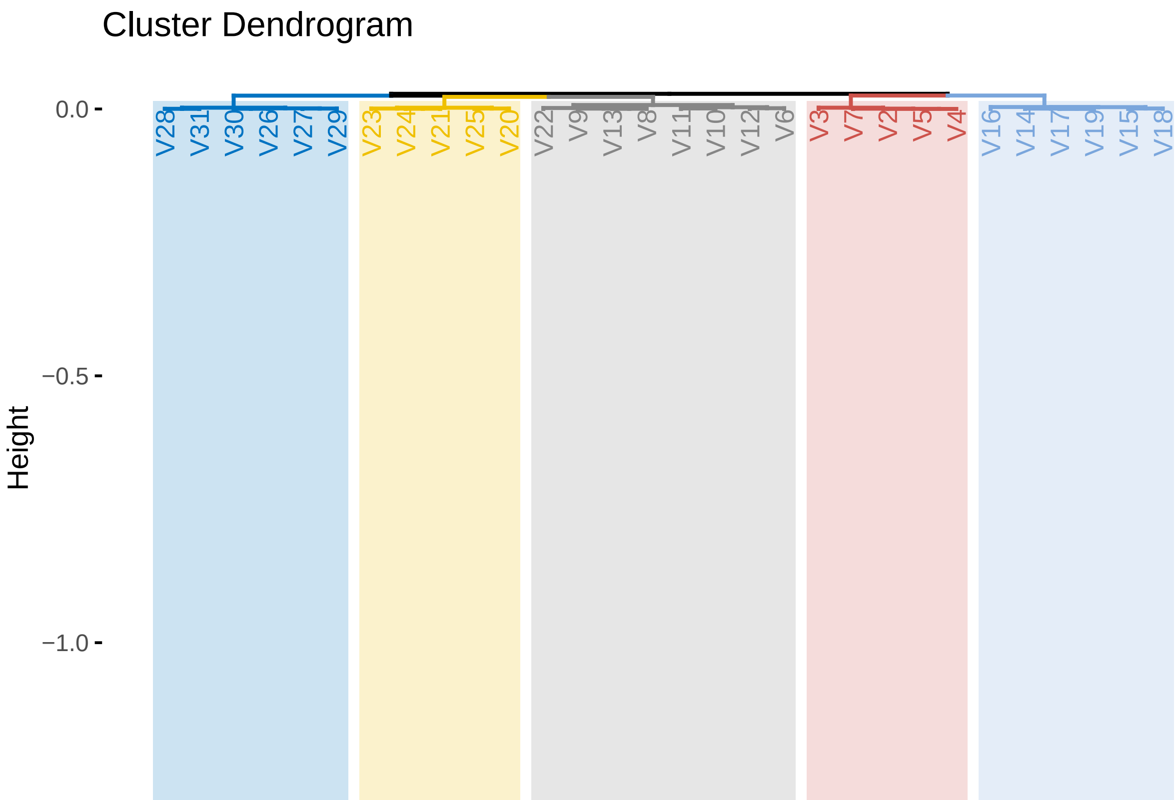


\#\# Hierarchical Clustering

```r
res.hcpc <- HCPC(pca_data, graph = FALSE)
fviz_dend(res.hcpc,
          cex = 0.7,                     # Label size
          palette = "jco",               # Color palette see ?ggpubr::ggpar
          rect = TRUE, rect_fill = TRUE, # Add rectangle around groups
          rect_border = "jco",           # Rectangle color
          labels_track_height = 0.8    # Augment the room for label
          )
```

```
## Registered S3 method overwritten by 'dendextend':
##   method     from
##   rev.hclust vegan
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## i The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <https://github.com/kassambara/factoextra/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


Cluster Dendrogram
Height
0.0
−0.5
−1.0
V28
V31
V30
V26
V27
V29
V23
V24
V21
V25
V20
V22
V9
V13
V8
V11
V10
V12
V6
V3
V7
V2
V5
V4
V16
V14
V17
V19
V15
V18